Relatório de CT213

Aluno: Henrique Fernandes Feitosa

1. Implementação da máquina de estados finitos

A máquina de estados escolhida para a implementação é a que é apresentada na Figura 1.

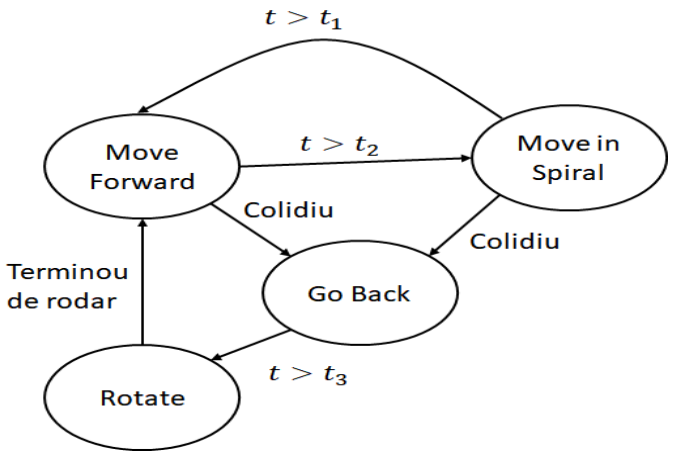

Figura 1: máquina de estados finita do comportamento do Roomba.

Inicialmente, buscou-se realizar a implementação do estado Move Forward. Para isso, criou-se uma variável denominada 'cont' que seria responsável por demarcar o tempo. Na função para checar a transição de estados, foram analisadas duas possibilidades, se o robô batesse na parede e se o tempo limite fosse excedido. Caso o robô colidisse com a parede, ele iria pro estado 'Go Back', caso o tempo limite fosse excedido ele iria para o estado 'Move in Spiral'. Na Figura 2, encontra-se um exemplo onde pode-se encontrar as duas possíveis transições de estados.

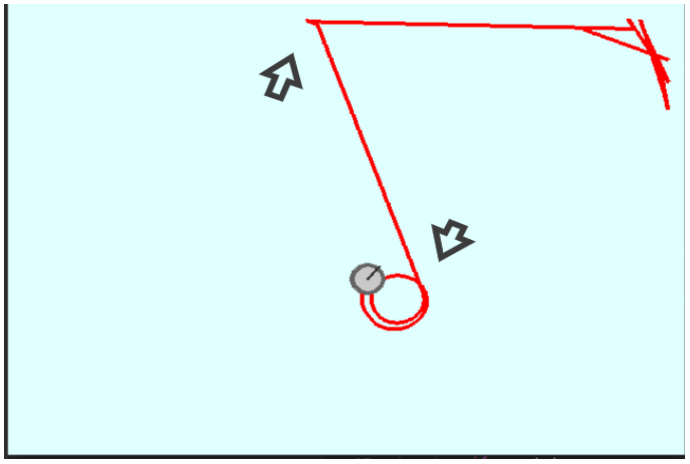

Figura 2: Figura que mostra as duas possíveis transições de estado.

Também pode-se observar na figura o funcionamento do estado 'Go Back'. Assim que o robô colide na parede, ele vai para trás por um determinado período de tempo, o qual foi marcado por uma variável chamada de 'count'. Após isso, quando o tempo limite foi excedido, ele vai para o estado 'RotateState'.

Nesse estado, seleciona-se um número aleatório no intervalo de [-4,4]. Após isso, o módulo do número e extraído para servir como tempo de execução, o qual vai ser contado pela variável de nome ‘count2’. Assim, temos duas possibilidades, se o número selecionado for positivo, ele gira no sentido horário, caso contrário, no sentido anti-horário. Nos dois casos, o próximo estado é o ‘MoveFoward’, assim como mostrado na Figura 2.

Por fim, resta o estado ‘MoveinSpiral’. Para isso, foi usado uma variável chamada de ‘contador’. Assim, temos duas possibilidades de transição, se o tempo limite for excedido, ele vai para o estado ‘MoveFowardState’ e se ele colidir com uma parede, ele vai para o estado ‘GoBackState’. A Figura 3 mostra a transição entre esses dois estados.

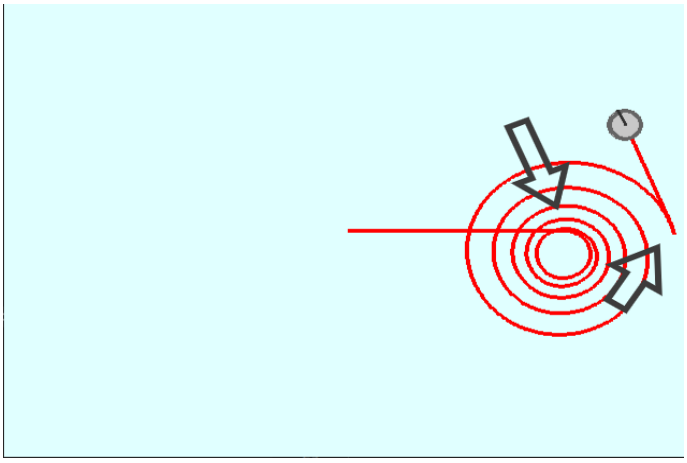

Figura 3: Figura que mostra as duas possíveis transições de estado.

2. Implementação da árvore de comportamentos

Para a implementação da árvore de comportamento, criou-se dois nós do tipo sequence, ao primeiro deles foram adicionados dois filhos, ‘MoveForwardState’ e ‘MoveinSpiral’ , no segundo foram adicionados o ‘GoBackState’ e o ‘RotateState’. Em seguida, criou-se outro nó do tipo selector e foram colocados os dois nós do tipo sequence como seus filhos. Por fim, a árvore de comportamentos ficou como a representada na Figura 4.

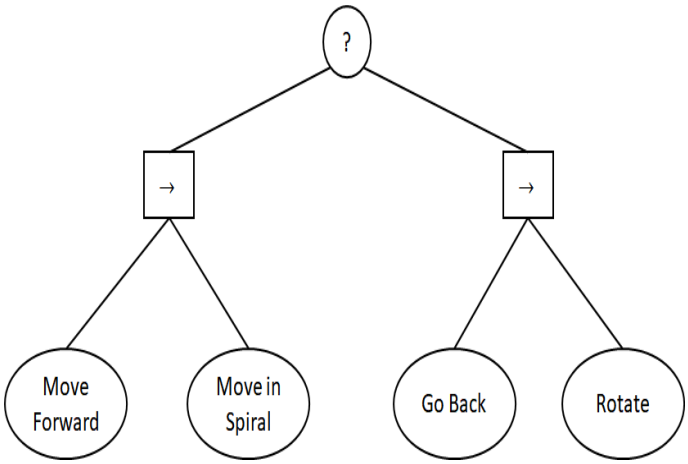

Figura 4: *behavior tree* do comportamento do Roomba.

A lógica de ações para implementar os comportamento foram iguais em todos os métodos, o único fator que mudou foi a lógica de transição que será explicada abaixo.

No estado ‘MoveFowardState’, o método execute retornava running e chamava o método enter a cada ciclo em que o tempo não ultrapasse o limite. Se o tempo ultrapassase o limite, o método retornaria sucess e o robô iria para o nó ‘MoveinSpiral’ , caso o robô colidisse, retornaria failure e o robô iria para o nó ‘GoBackState’. A Figura 5 mostra a transição entre esses estados.

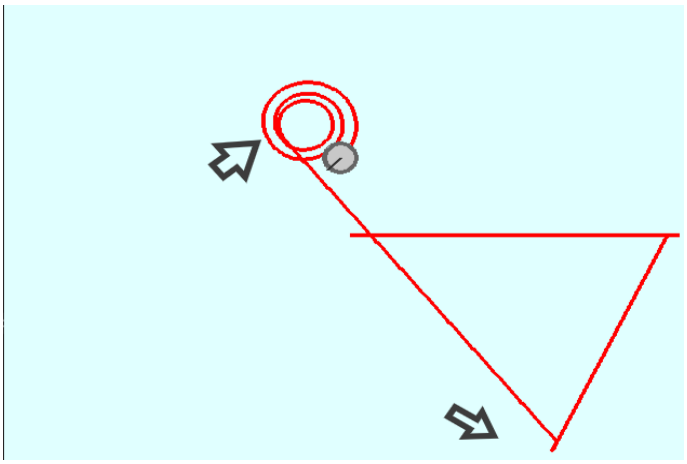

Figura 5: Figura que mostra as duas possíveis transições de estado.

Caso ele vá para o nó ‘GoBackState’, ele vai para trás por um certo tempo. Caso esse tempo seja excedido, o método retorna sucess e o robô vai para o nô rotate, se o tempo ainda não tiver sido excedido, o método retorna running. Após esse método retornar sucess, o robô vai para o nó ‘RotateSate’, que retorna running se o tempo não for excedido e sucess quando esse tempo for ultrapassado.

Por fim, resta discutir sobre o método ‘MoveinSpiral’. Caso o robô colida com a parede, o método retorna failure, caso ele não colida, o métdo irá retornar sucess se o tempo limite for excedido e running se ele não for. A figura 6 mostras as duas possíveis transições de estados.

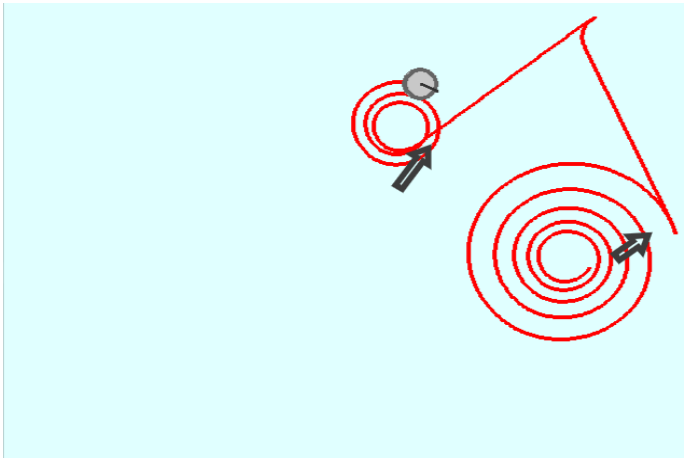

Figura 6: Figura que mostra as duas possíveis transições de estado.